Representation:

$x^{<t>}, y^{<t>}$ → t^{th} element in sequence

T_x, T_y = length of sequence

$x^{(i)<t>}$ → for i^{th} training example

$T_x^{(i)}, T_y^{(i)}$ → length of i^{th} training sequence

can change per training example

Vocabulary / dictionary: store all words

then use one hot representation

having a vector per word full of zeros except ith location where i is the position of word in dictionary

$x^{<t>}$ is a one hot vector.

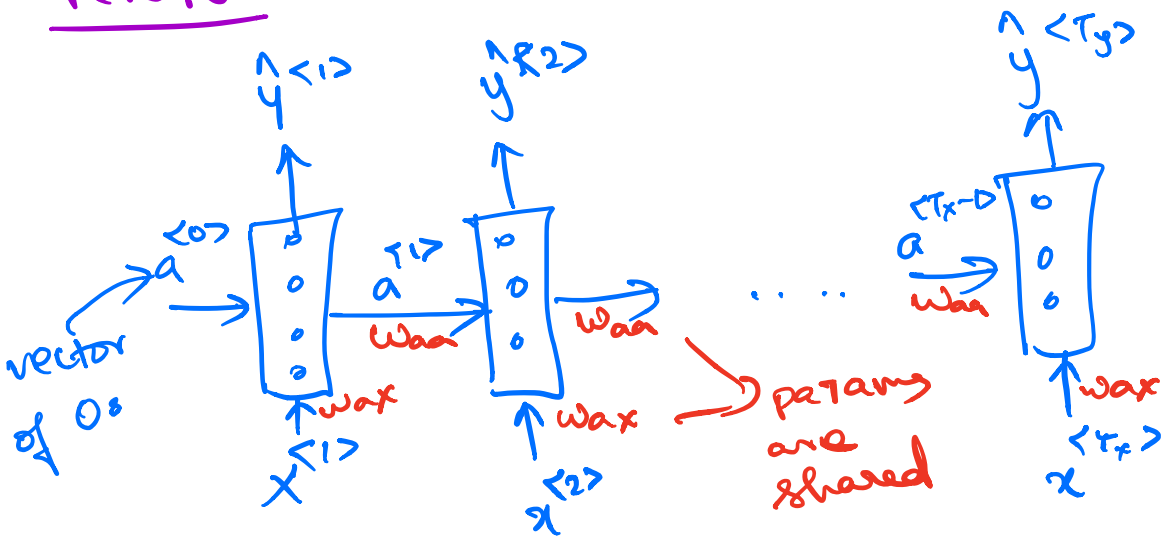
$$x^{<t>} = \begin{cases} 1, & \text{for } i = \text{dictionary[word]} \\ 0, & \text{elsewhere} \end{cases}$$

we cannot use standard NN:

i) i/p and o/p can be of different lngth

ii) doesnt share features learned from different parts of the text.

RNN:



But this RNN doesn't consider words after it. (ie) it processes sentences only from left to right

forward prop:

$$a^{<0>} = \vec{0}$$

$$a^{<1>} = g(w_{aa} a^{<0>} + w_{ax} x^{<1>} + b_a)$$

tanh/relu

$$\hat{y}^{<1>} = g(w_{ya} a^{<1>} + b_y)$$

sigmoid/softmax

$$a^{<t>} = g(w_{aa} a^{<t-1>} + w_{ax} x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(w_{ya} a^{<t>} + b_y)$$

$$a^{<t>} = g(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$$

W_{aa}: (100, 100), $a^{<t-1>}$: 100, W_{ax}: (100, 10,000), $x^{<t>}$: 10,000

$$\hat{y}^{<t>} = g(W_{ya} a^{<t>} + b_y)$$

$$\hat{y}^{<t>} = g(w_y a^{<t>} + b_y)$$

$$a^{<t>} = g(w_a [a^{<t-1>}, x^{<t>}] + b_a)$$

$[w_{aa}; w_{ax}]$: 100 × (100 + 10000) $= w_a$: (100, 10100)

$[a^{<t-1>}, x^{<t>}]$: 10100

$[w_{aa}; w_{ax}] \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = w_{aa} a^{<t-1>} + w_{ax} x^{<t>}$

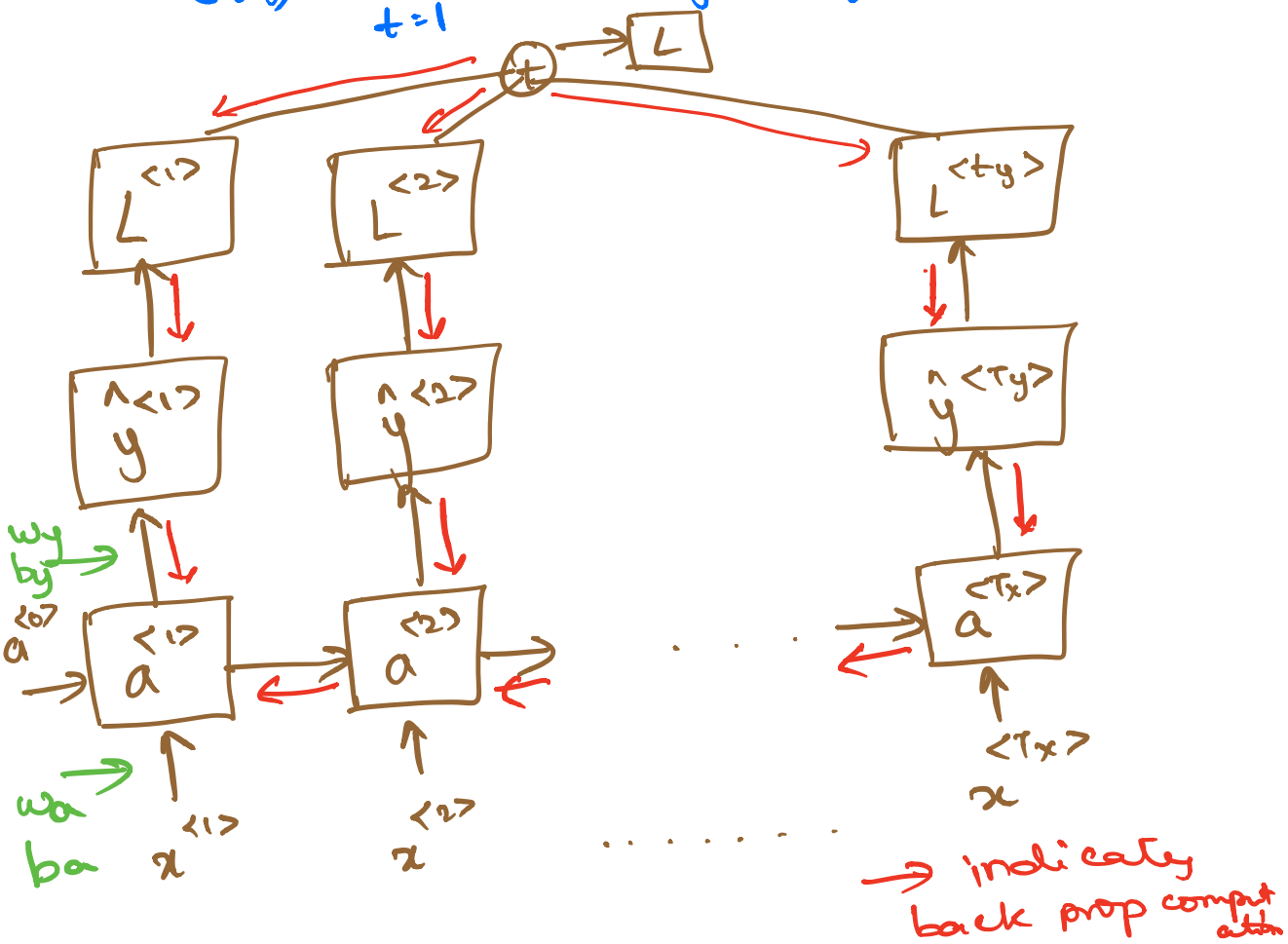
Back prop:

Backprop through time

cross entropy loss

$$L^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}) = -y^{\langle t \rangle} \log \hat{y}^{\langle t \rangle} - (1 - y^{\langle t \rangle}) \log (1 - \hat{y}^{\langle t \rangle})$$

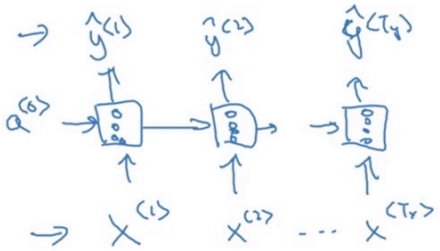
$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$



we can also have data where

$T_x \neq T_y$

$T_x = T_y$

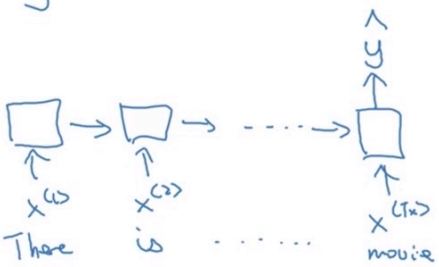


Many-to-many

Sentiment classification

x = text

y = 0/1 ... 5

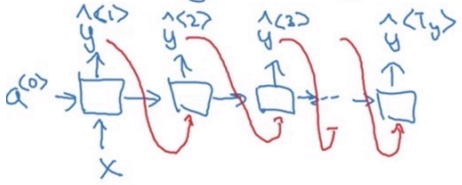


Many-to-one

One-to-one

Music generation

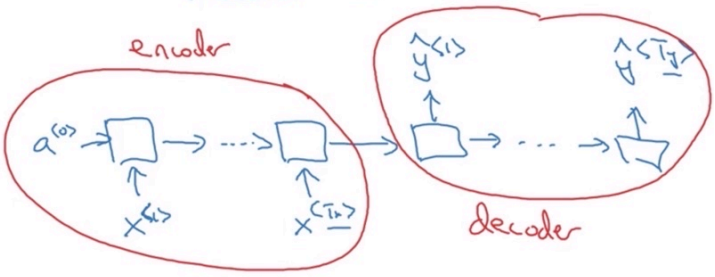
$x \rightarrow y^{<1>} y^{<2>} \ldots y^{<T_y>}$



One-to-many

$x = \phi$

Machine translation



Many-to-many ($T_x \neq T_y$)

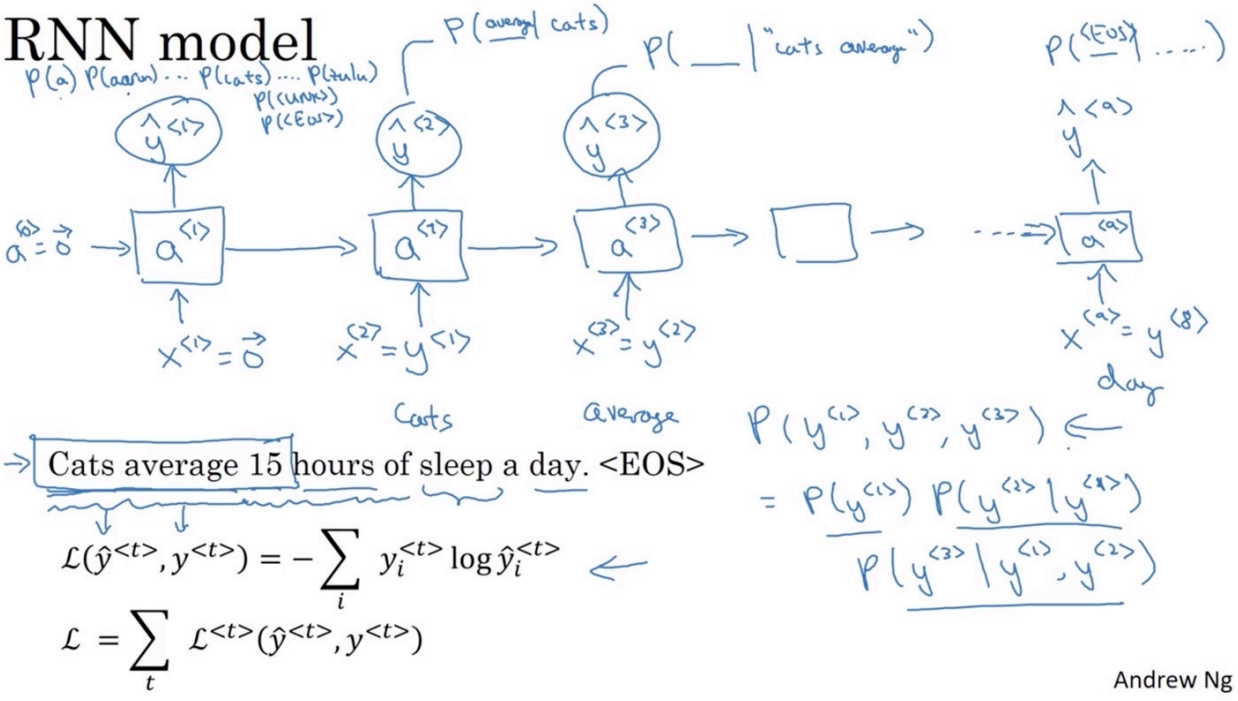
Language model:

sentence $\rightarrow$ P(sentence)

Training set: large corpus of english text.

input → tokenize (add <EOS>)
↓
get $y^{<t>}$ ($x^{<t>} = y^{<t-1>}$)

for unknown word use <UNK>

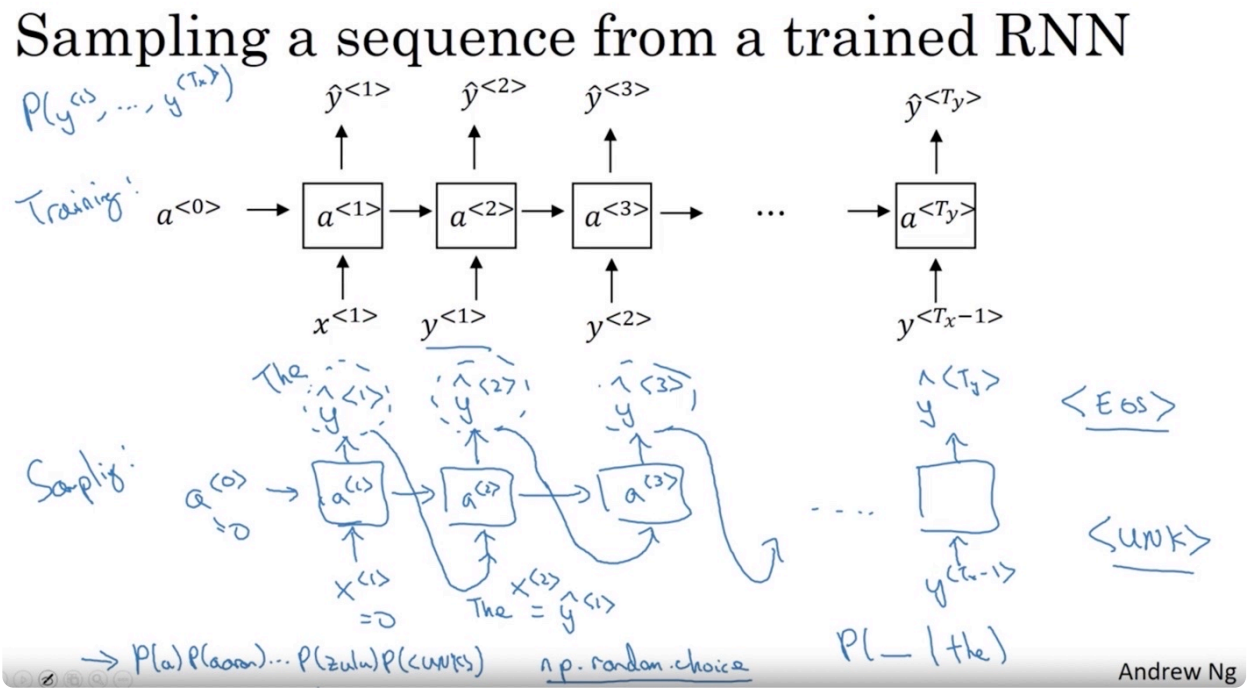


we can also sample sentences from this network using random sampling

To stop:

i) sample 10-... words

ii) add <EOS> to vocabulary and stop when <EOS> is sampled.



we can also build character level language model (instead of word level) in which case vocab will have alphabets instead of words

In character level → we dont have to worry about unknown words

but we need longer sequences and hence performs less when capturing long term dependencies

In RNNs, we have the problem of vanishing gradients.

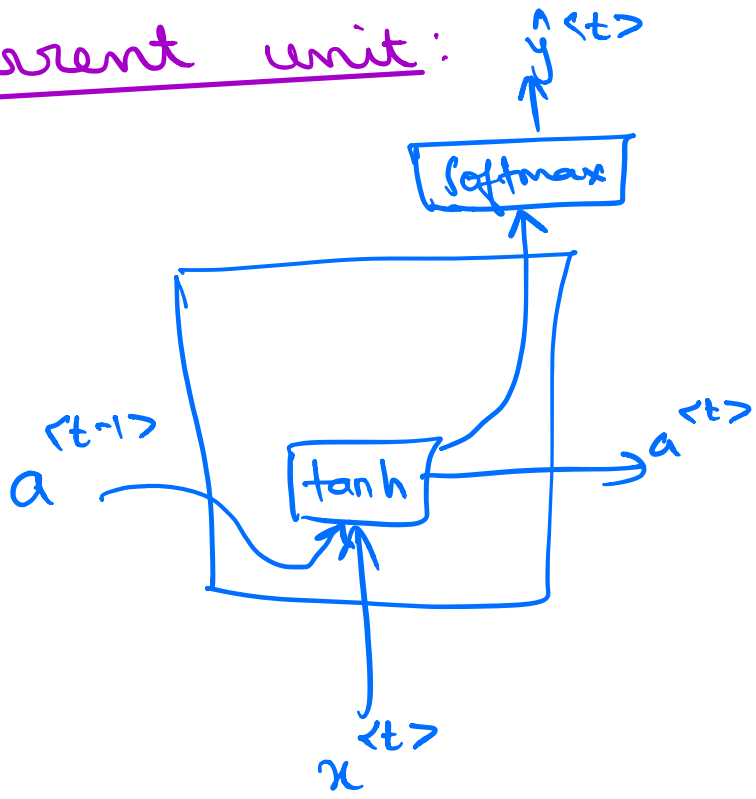
so, $y^{<100>}$ tends to be strongly influenced by few previous $y^{<t>}$ but not by $y^{<t>}$'s earlier in the sequence.

(ie) RNNs fail to capture long range dependencies.

for exploding gradients: apply gradient clipping.

Gated recurrent unit:

RNN unit:



GRU:

c: memory cell

$c^{<t>} = a^{<t>}$

candidate to replace $c^{<t>}$ → $\tilde{c}^{<t>} = \tanh\left(W_c\left[c^{<t-1>}, x^{<t>}\right] + b_c\right)$

we have a gate: Γ_u (gamma) (update gate)

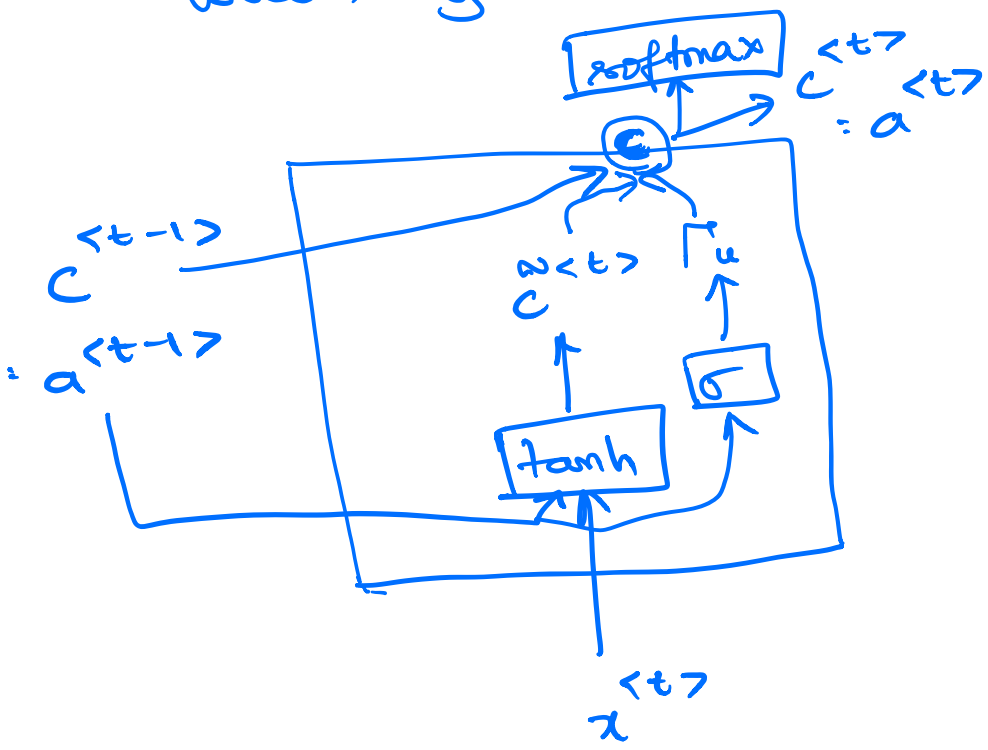
mostly 0/1 → $\Gamma_u = \sigma\left(W_u\left[c^{<t-1>}, x^{<t>}\right] + b_u\right)$

when to update c + what to update

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$$

set $c^{<t>} = \tilde{c}^{<t>}$ when gate = 1

when gate = 0, $c^{<t>} = c^{<t-1>}$



as Γ_u will be close to zero, $c^{<t>} = c^{<t-1>}$, this GRU wont suffer with vanishing gradient

Full GRU:

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

Γ_r: relevance of $c^{\langle t-1 \rangle}$ to $c^{\langle t \rangle}$

$$\Gamma_u = \sigma\left(W_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

$$\Gamma_u = \sigma\left(W_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right)$$

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) * c^{\langle t-1 \rangle}$$

$$a^{\langle t \rangle} = c^{\langle t \rangle}$$

Long short term memory: (LSTM)

no longer the case that $a^{\langle t \rangle} = c^{\langle t \rangle}$

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

we have two update gate

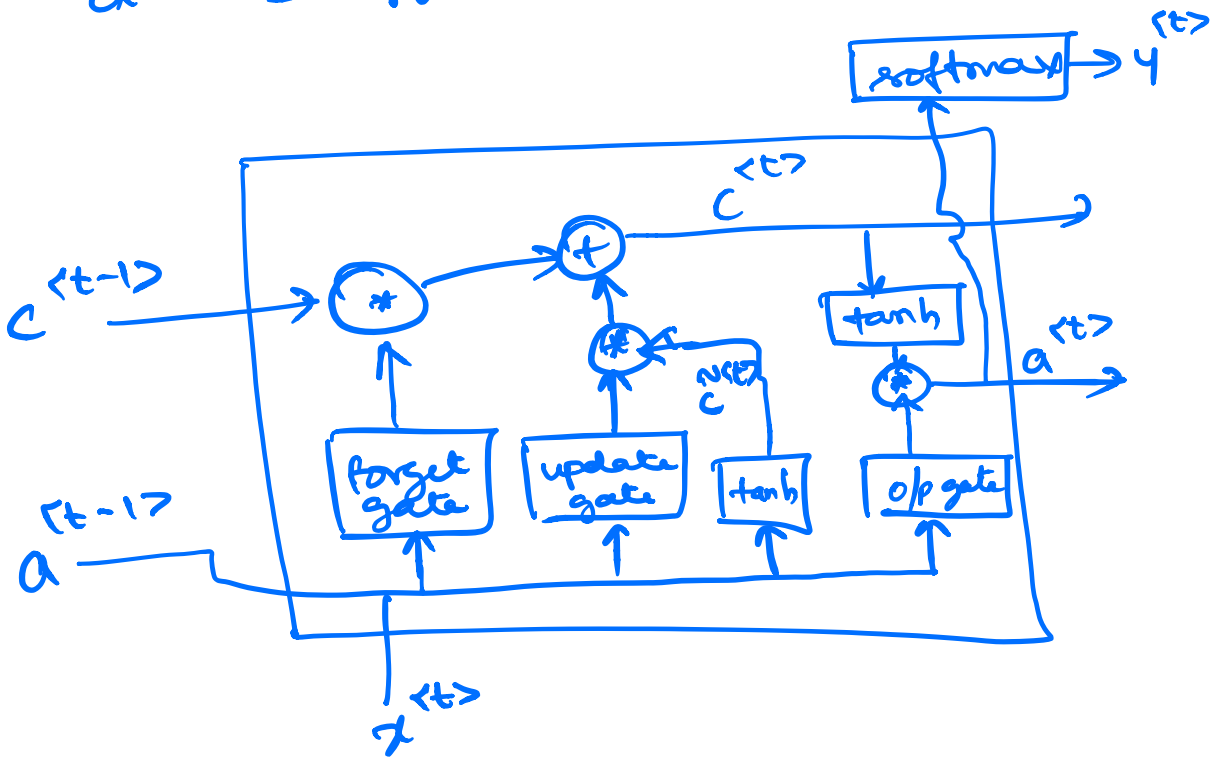
update gate $\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$

forget gate $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$

output gate $\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$



If update / forget gate works correctly, $c^{\langle 100 \rangle}$ can be $c^{\langle 1 \rangle}$.

so this helps in overcoming vanishing gradients.

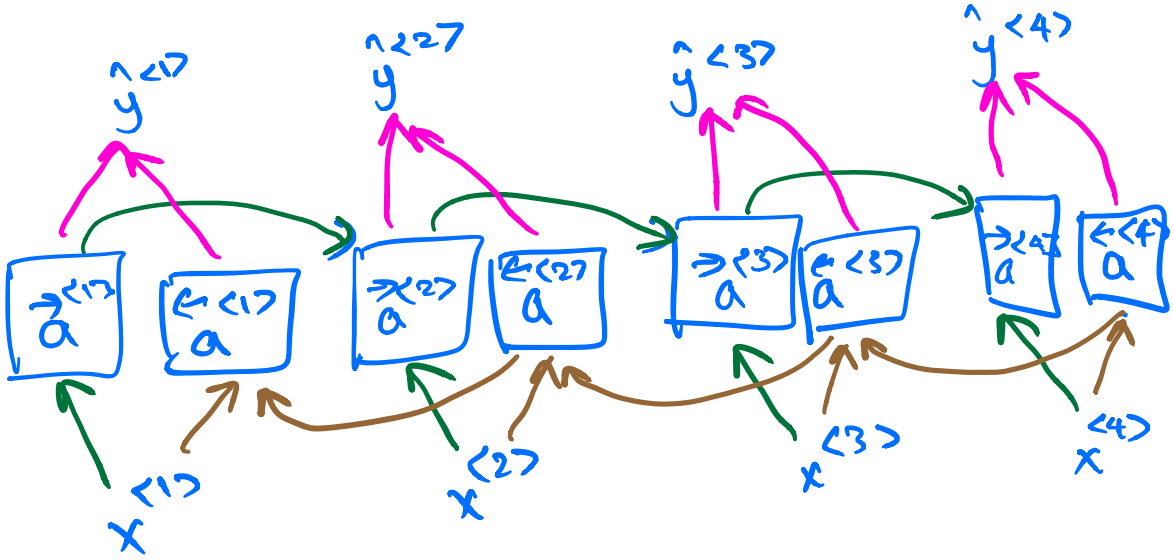
variation:

Instead of just having gate values depend on $a^{\langle t-1 \rangle}, x^{\langle t \rangle}$ we can add $c^{\langle t-1 \rangle}$. This is called peephole connection

GRU → computationally effective

LSTM → much effective

Bidirectional RNN:



acyclic graph is formed

→ step 1
→ step 2
→ step 3

$$\hat{y}^{<t>} = g(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

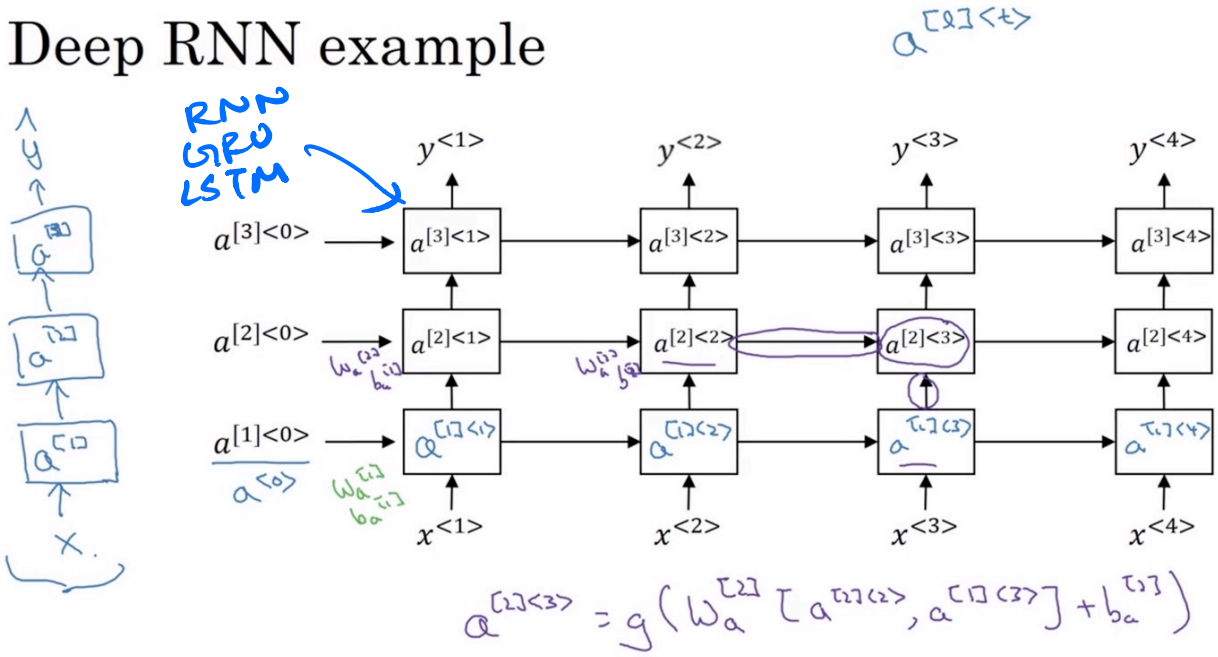
each box can be basic RNN block or GRU or LSTM block

but we need entire sequence before processing

Deep RNN:

even 3 layers stacked can take significant time.

Deep RNN example



we can also take $y^{<t>}$ used as input for further DNN without horizontal connection.

Word representation:

O_x → one hot vector with x^{th} position as 1.

This doesn't know the relation between words. Always O_x and O_y will be zero.

featurized representation:
word embedding

This will learn values for each feature for the word. Similar words will have somewhat common features

Featurized representation: word embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
⋮	⋮	⋮				
size						
cost						
alive						
verb						

300

e_{5391} e_{9853}

I want a glass of orange juice.

I want a glass of apple juice.

Andrew Ng

using t-SNE to plot the embedding in 2D, we can see that more related words are closer together.

(ie) orange and apple are closer than man and apple.

∴ we can generalize better in test data.

Transfer learning and word embeddings

1. Learn word embeddings from large text corpus. (1-100B words)

   (Or download pre-trained embedding online.)

2. Transfer embedding to new task with smaller training set. (say, 100k words) → 10,000 → 300

3. Optional: Continue to finetune the word embeddings with new data.

In face encoding → we expect to input unknown face to network and it still gives us encoding.

In word embedding → we have fixed vocabulary and learn encoding for the words.

Analogies

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

e_{5391} e_{man} e_{woman}

$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Man → Woman as King → ? Queen

$e_{man} - e_{woman} \approx e_{king} - e_?$, $? \approx$ Queen

$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

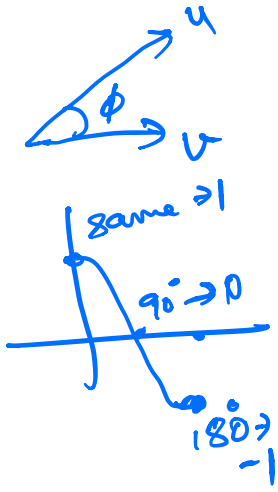
find word w:

$$\underset{w}{\text{argmax}} \; \text{similarity}(e_w, e_{king} - e_{man} + e_{woman})$$

most common similarity:

cosine similarity:

$$\text{sim}(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$



$e_b - e_a \approx e_w - e_c$

$e_c + e_b - e_a \approx e_w$

Neural language model: (to learn embedding)

1) construct one hot vector for each word

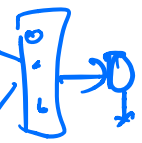
2) $E * O_x \rightarrow e_x$ for all words

3) give e_x for necessary words as an input to a neural network that has softmax final layer to predict next word.

instead of all words we can also have fixed historical data like last 4 words

$O_x \rightarrow E \rightarrow e_x$

$O_y \rightarrow E \rightarrow e_y$



parameters $\Rightarrow$ E, weights of neural network

use gradient descent to find E.

Other context/target pairs

I want a glass of orange juice to go along with my cereal.

context: glass of orange

target: juice

Context: Last 4 words. → good for language model

- 4 words on left & right — a glass of orange ? to go along with
- Last 1 word — orange ?
- Nearby 1 word — glass ?

(good to learn word embedding)

skip gram

Skip gram:

* context - target pair for training
  - context: randomly chosen in a window
  - target

$x \rightarrow y$

context $c \rightarrow$ target t

$O_c \rightarrow E \rightarrow e_c \rightarrow$ softmax $\rightarrow \hat{y}$

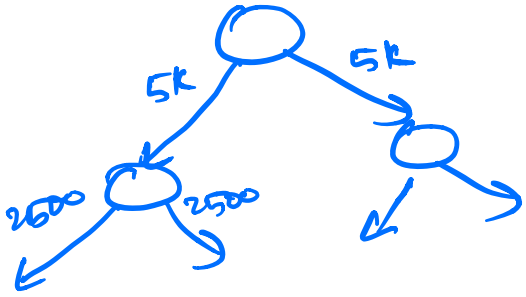
$$P(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10000} e^{\theta_j^T e_c}}$$

($\theta_t^T e_c$: target)

$$L(\hat{y}, y) = -\sum_{i=1}^{10000} y_i \log \hat{y}_i$$

The summation in $P(t|c)$ is computationally expensive

hierarchical softmax



https://www.youtube.com/watch?v=pzyIWCelt_E

In practice, we have unbalanced tree to have common words are in high level.

selecting $c \rightarrow$ uniformly random with some heuristics to select uncommon words.

selecting $t \rightarrow$ random word in a window.

New learning problem:

context - word → is this context target pair

x → y

context → randomly from sentence

target word → word in some the window, + ve example

every the sample, we have k - ve. → random word from dictionary, - ve example

model:

x → is target (y)

(context, word)

c, w

$$P(y=1 \mid c,t) = \sigma(\theta_t^T e_c)$$

$O_{*c} \rightarrow E \rightarrow e_c$

binary classifi

each word → is target or not

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum f(w_i)^{3/4}}$$

Glove: global vector for word representation.

X_{ij} = # times j appears in context of i

minimize:

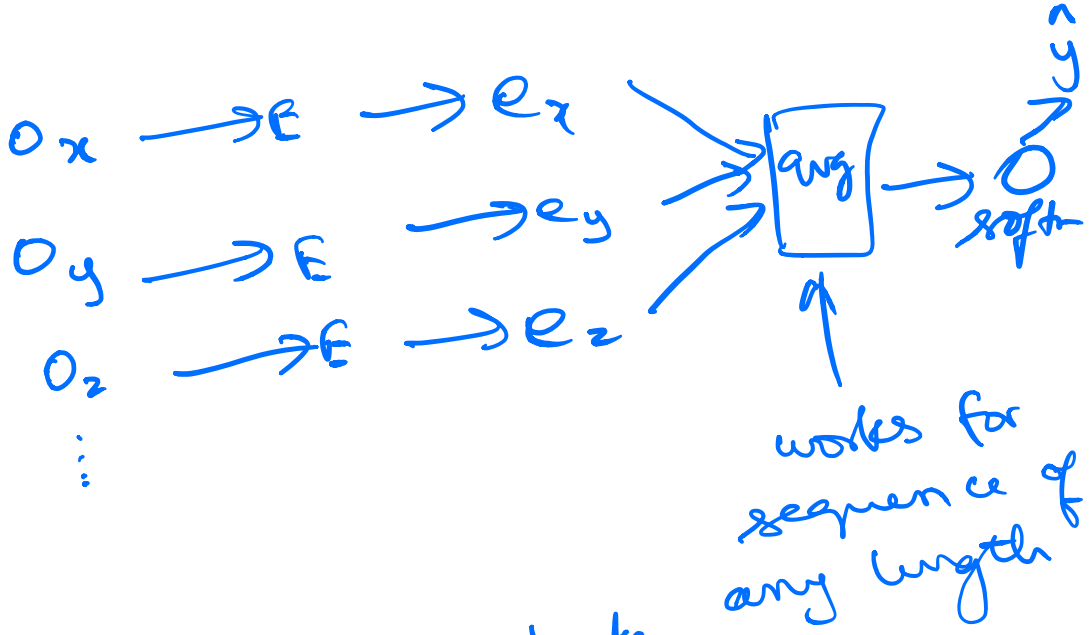
$$\sum_{i=1}^{10000} \sum_{j=1}^{10000} f(X_{ij}) \left(\theta_t^T e_c + b_t + b_c - \log X_{ij} \right)^2$$

weighting term

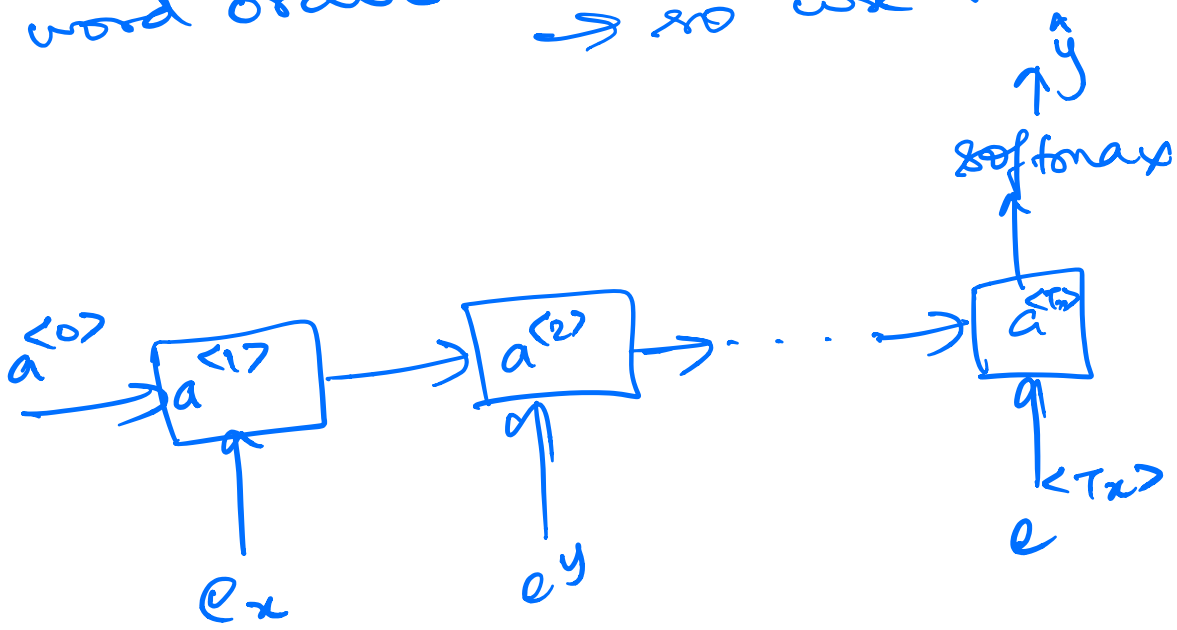
$= 0$ when $X_{ij} = 0$ so that we won't have to calculate $\log(0)$

Since θ_T and e_j are symmetric. So initialize randomly and $e_w^{(final)} = \frac{e_w + \theta_w}{2}$

Sentiment classification:



But this doesn't take word order into consideration
→ so we rnn

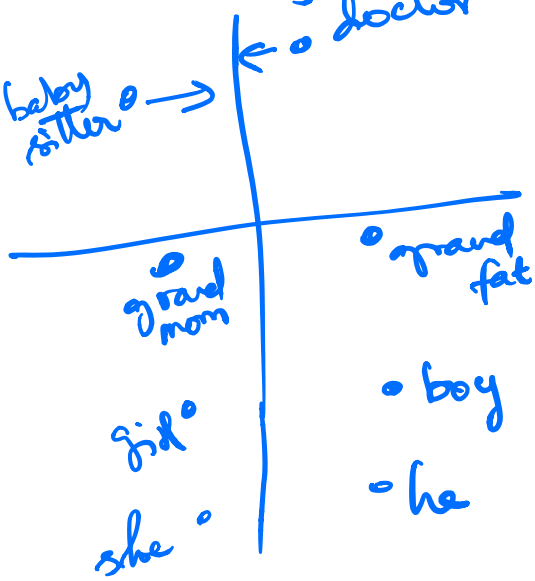


Debiasing word embedding:

man : programmer :: women : homemaker

(😵 wrong)

embedding:



1. identify bias direction

$e_{he} - e_{she}$

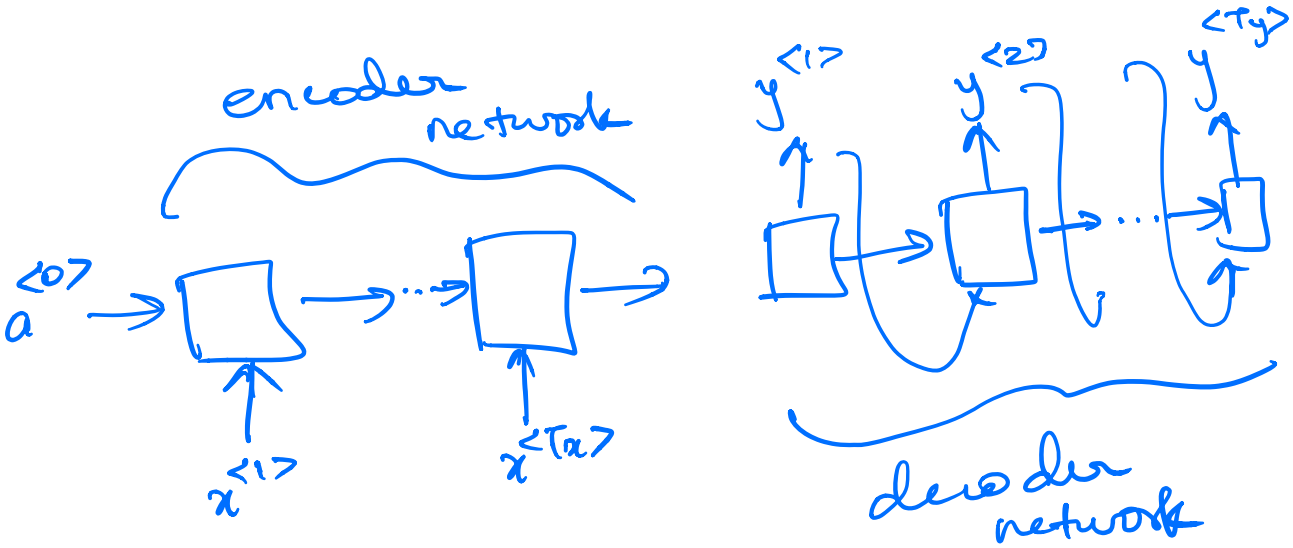
2. neutralize: for word that is not definitional (doctor), project to get rid of bias

to maintain distance between the "baby sitter" and "grand mom/dad" same

3. equalize pairs (boy, girl)

so more boy/girl to be same distance to that word

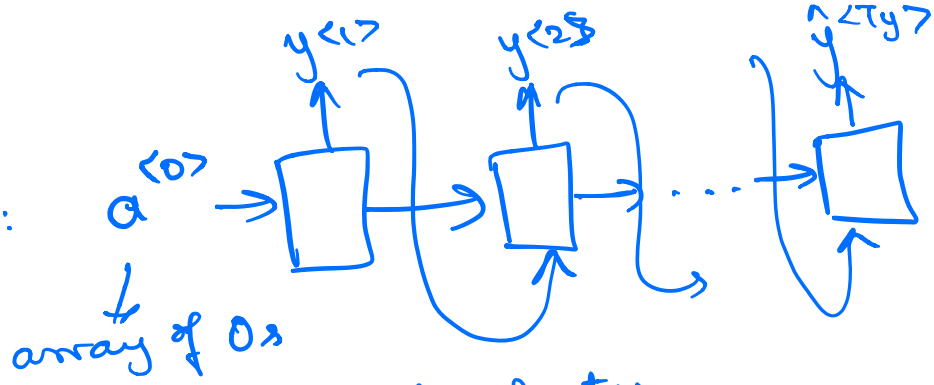
Sequence to sequence model:



works well when enough dataset is given.

For image captioning, use a pretrained CNN as encoder network
↳ without last layer

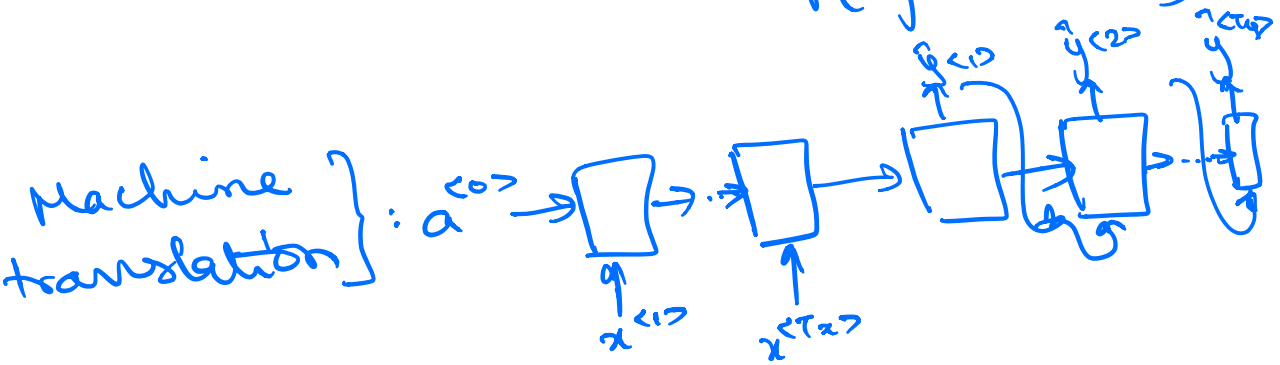
Language model:



array of 0s

this calculates $P(y^{<1>} \cdots y^{<T_y>})$

Machine translation:



here the decoder part is same as the one we used for language model.

Except instead of $a^{<0>} = \vec{0}$, here we have $a^{<0>}$ as encoder network's output.

∴ here we calculate

$$P(\hat{y}^{<1>} \cdots \hat{y}^{<T_y>} \mid x^{<1>} \cdots x^{<T_x>})$$

here also called conditional language model

for french → English

$$P(y^{<1>} \dots y^{<T_y>} \mid x)$$

↑ english ↑ french

we will have probabilities for diff. english sentences.

(we cannot sample randomly from that)

So find $y^{<1>} \dots y^{<T_y>}$ that maximizes

$$P(y \mid x)$$

⇒ we use beam search

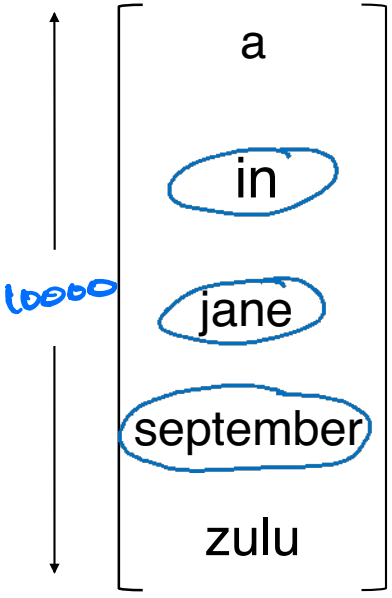
Greedy search (each word selected will have max prob)

⇒ overall optimality may not be maintained.

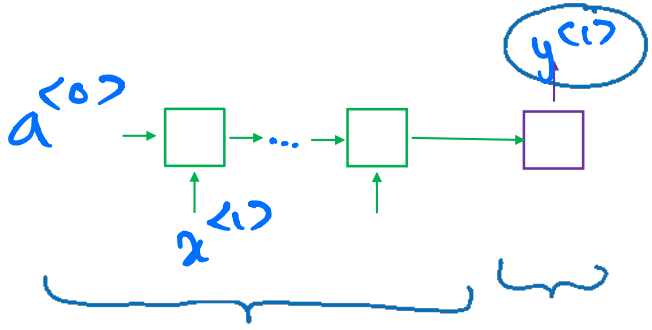
Beam search algorithm

$B = 3$ (beam width)

Step 1

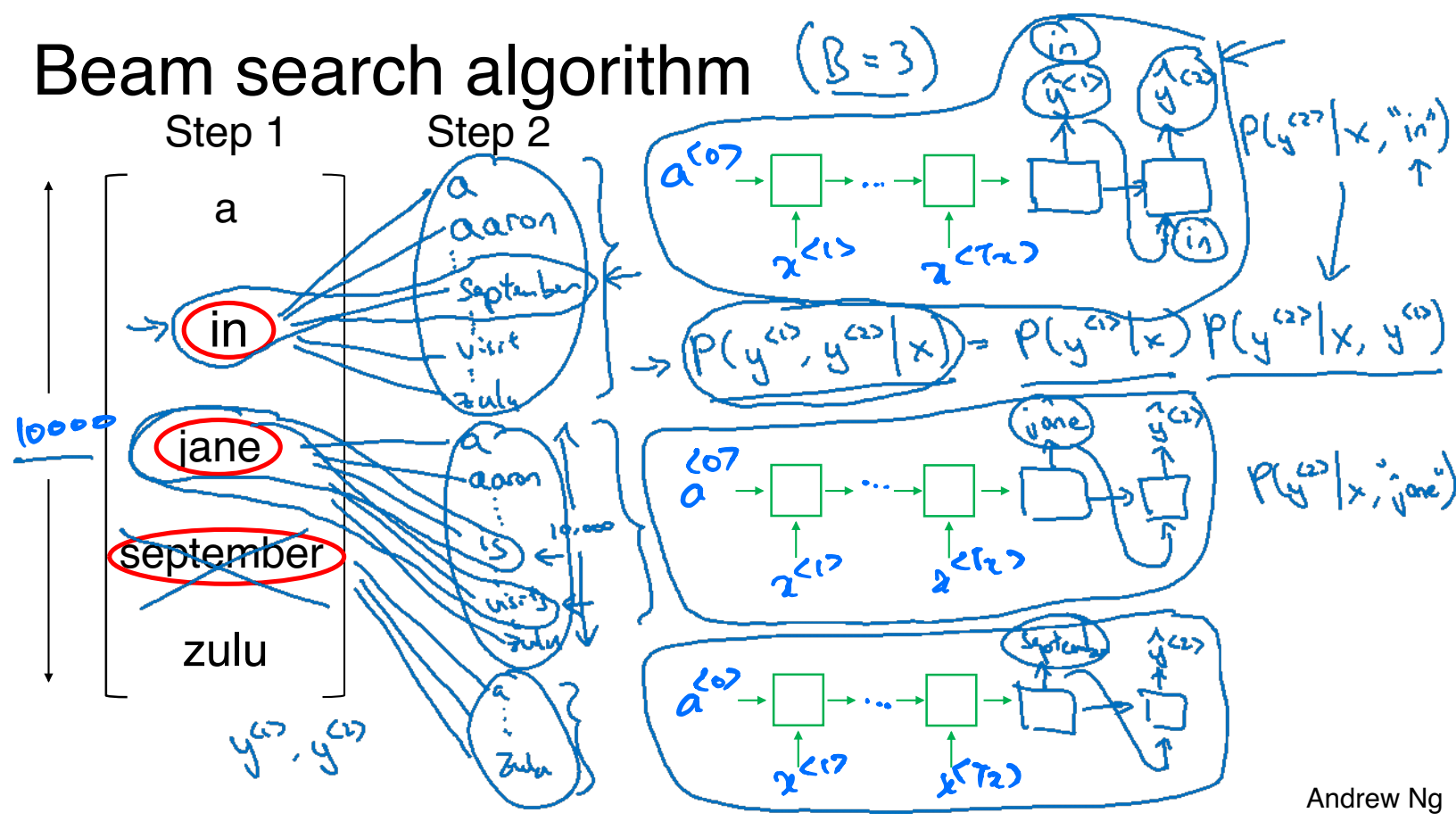


$\rightarrow P(y^{<1>} | x)$



B copies of network needed

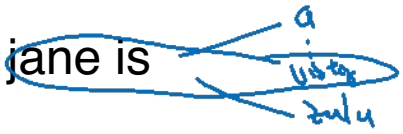
Beam search algorithm

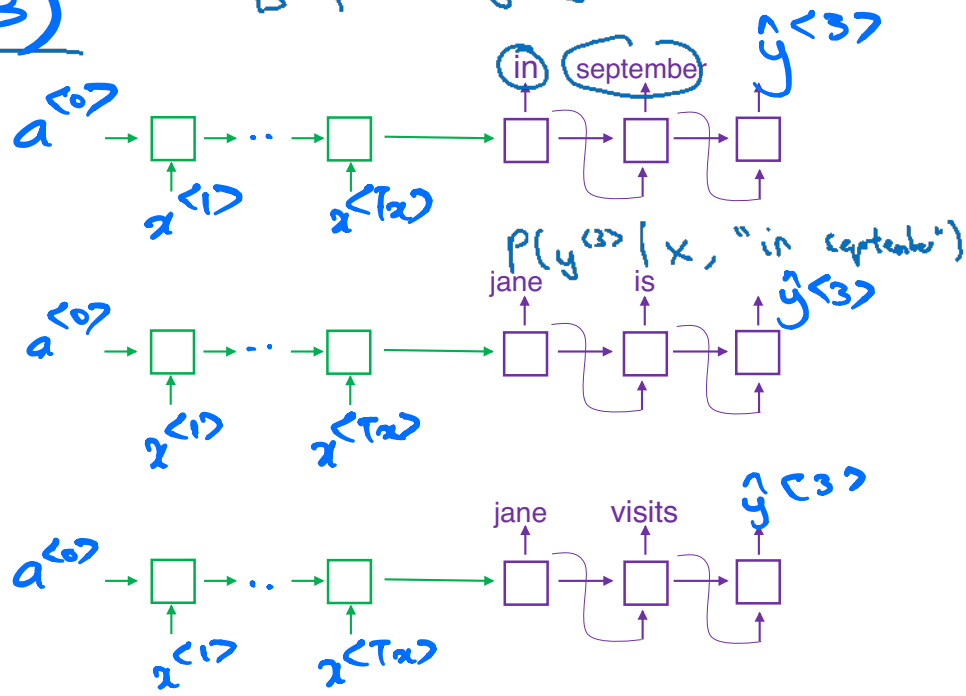


Andrew Ng

Beam search (B = 3)

B = 1 ⇝ greedy search





jane visits africa in september. <EOS>

Andrew Ng

length normalization:

we have:

$$\arg\max_{y} \prod_{t=1}^{T_y} P(y^{<t>} | x, y^{<1>} .. y^{<t-1>})$$

as these are small values, we may have underflow.

$$\therefore \arg\max_{y} \sum_{t=1}^{T_y} \log\left(P(y^{<t>} | x, y^{<1>} .. y^{<t-1>})\right)$$

This tends to prefer short output

so normalize it:

$$\arg\max_{y} \frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log\left(P(y^{<t>} | x, y^{<1>} .. y^{<t-1>})\right)$$

$\alpha = 1 \rightarrow$ full normalization

$\alpha = 0 \rightarrow$ no normalization

large B: better result (check many possib)
slower

small B: worse result
faster

unlike BFS/DFS, beam search runs faster but is not guaranteed to find exact maximum.

Error analysis:

statement, y^*, $\hat{y}$

y^*: human, $\hat{y}$: RNN + beam search

case 1: $P(y^*|x) > P(\hat{y}|x)$

$\Rightarrow$ beam search fault

case 2: $P(y^*|x) \leq P(\hat{y}|x)$

$\Rightarrow$ RNN fault

BLEU score: bilingual evaluation understudy

how good is the machine translation

If predicted statement ≈ reference stat

⇒ high bleu score

Give credit upto max. no. of times words occur in reference sentence

Bleu score on bigrams

Example: Reference 1: The cat is on the mat. ←

Reference 2: There is a cat on the mat. ←

MT output: The cat the cat on the mat. ←

	Count	$Count_{clip}$
the cat	2 ←	1 ←
cat the	1 ←	0
cat on	1 ←	1 ←
on the	1 ←	1 ←
the mat	1 ←	1 ←

↑

$\frac{4}{6}$

Bleu score on unigrams

Example: Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

→ MT output: The cat the cat on the mat. ($\hat{y}$)

$p_1, p_2, \ldots = 1.0$

$$p_1 = \frac{\sum_{\text{unigrams} \in \hat{y}} \text{Count}_{clip}(\text{unigram})}{\sum_{\text{unigram} \in \hat{y}} \text{Count}(\text{unigram})}$$

unigram

n-gram

$$p_n = \frac{\sum_{\text{n-grams} \in \hat{y}} \text{Count}_{clip}(\text{n-gram})}{\sum_{\text{n-grams} \in \hat{y}} \text{Count}(\text{n-gram})}$$

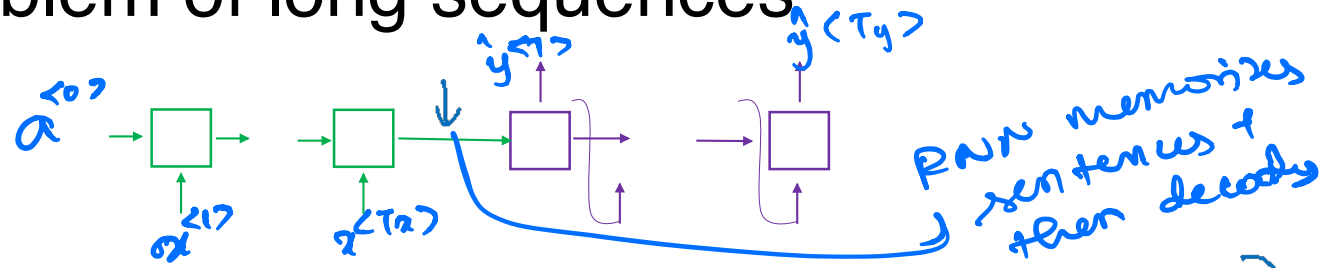
$$\left.\begin{matrix}\text{combined} \\ \text{bleu score}\end{matrix}\right\} = BP \exp\left(\frac{1}{4}\sum_{n=1}^{4} p_n\right)$$

BP ↓

brevity penalty → to penalize short sentences

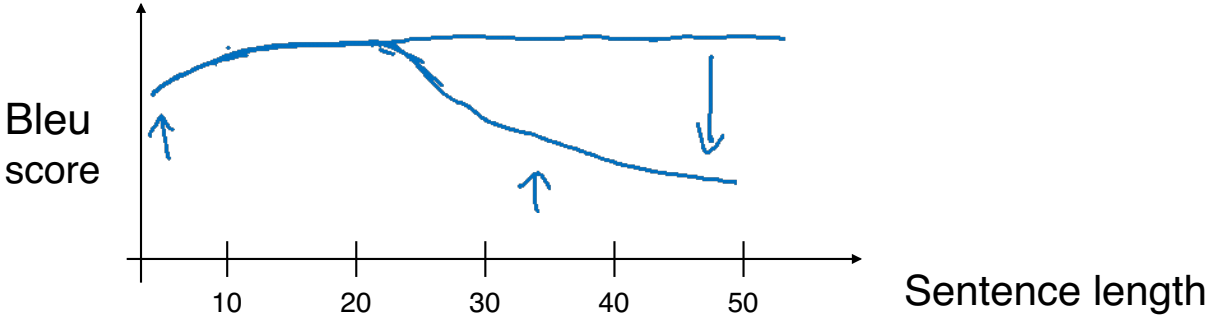
$$BP = \begin{cases} 1, & \text{if MT\_out\_len} > \text{reference\_out\_len} \\ \exp(1 - \text{ref\_o\_len}/\text{MT\_o\_len}), & \text{otherwise} \end{cases}$$

The problem of long sequences

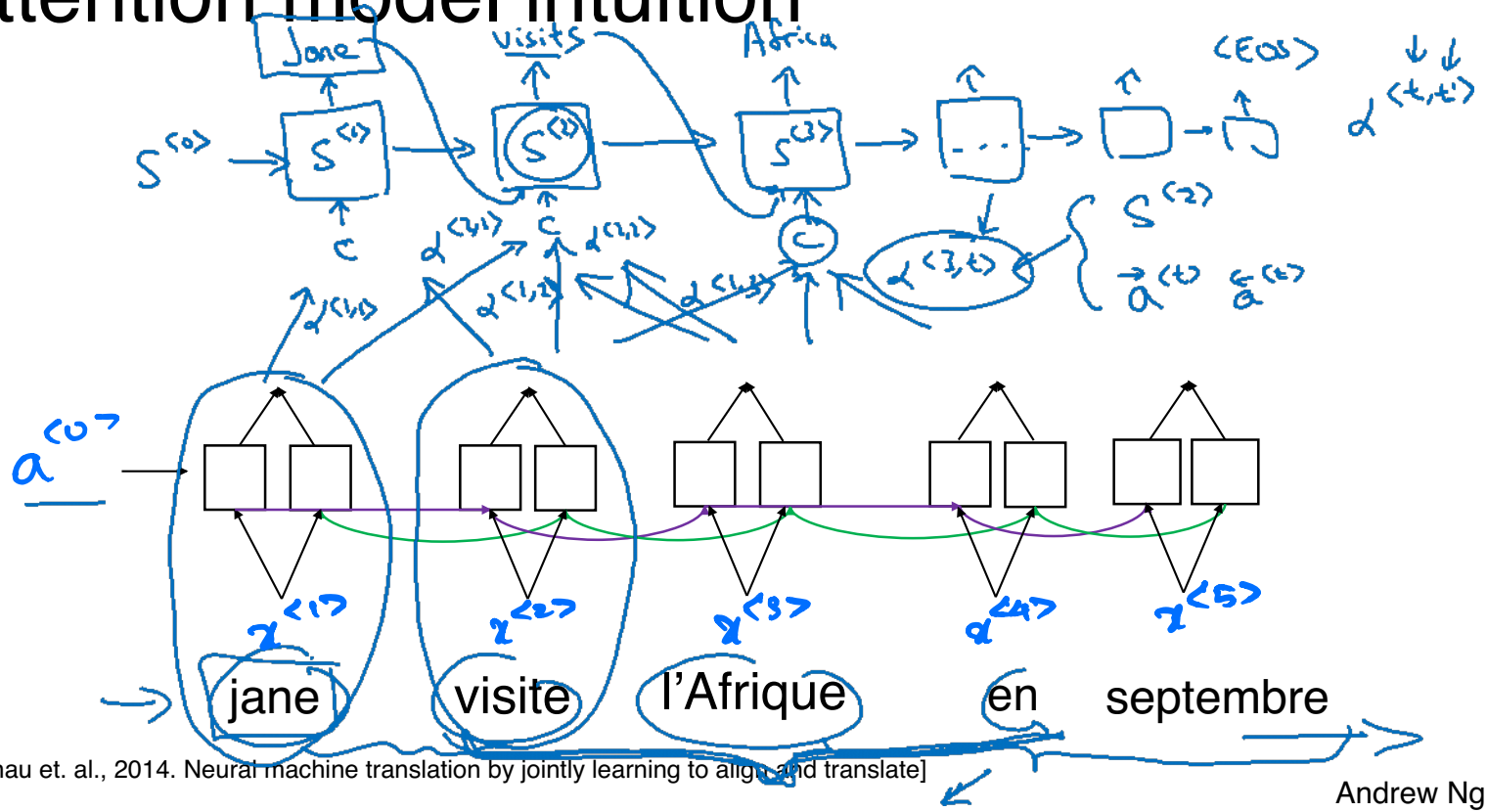


Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

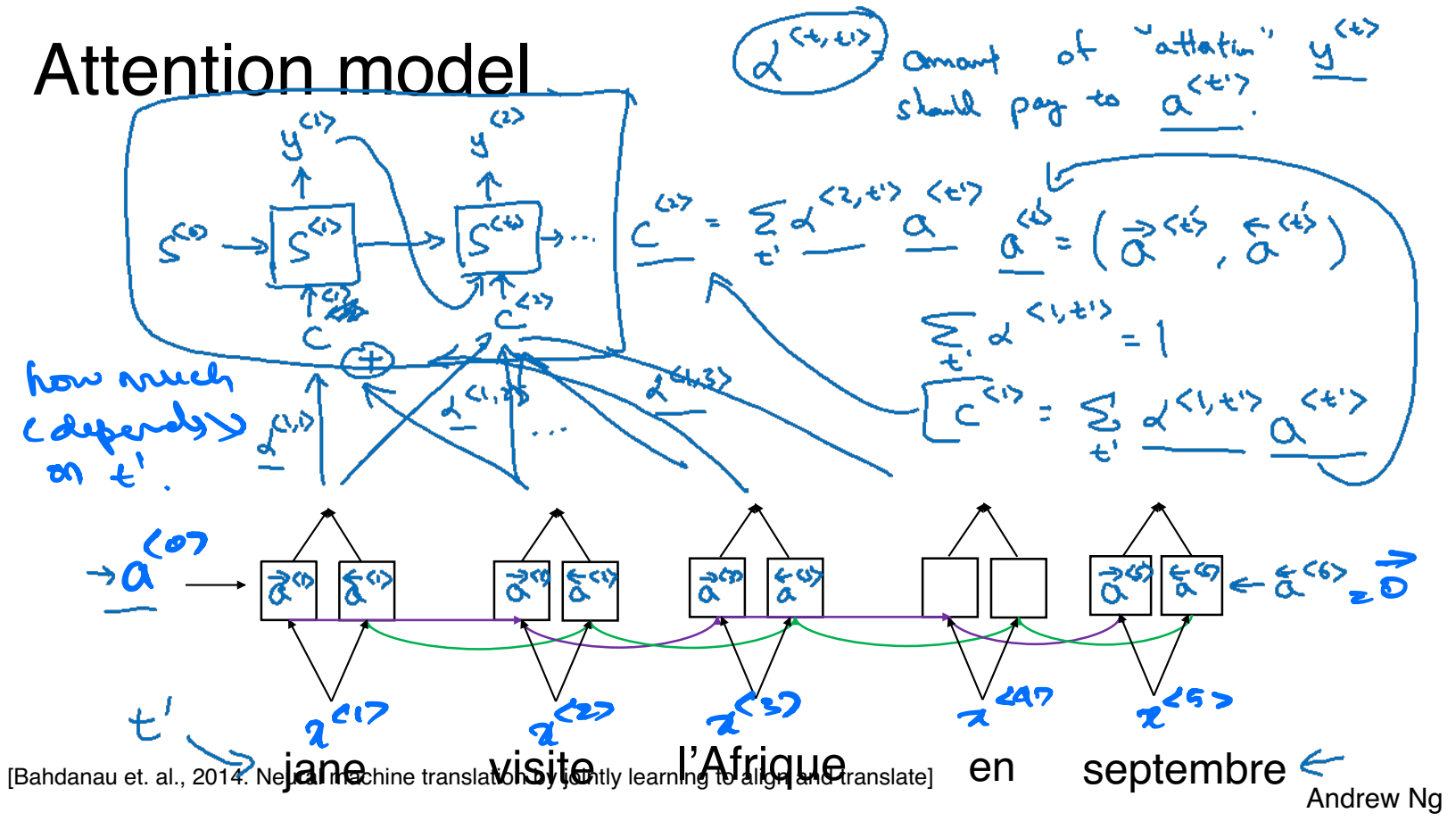


Attention model intuition



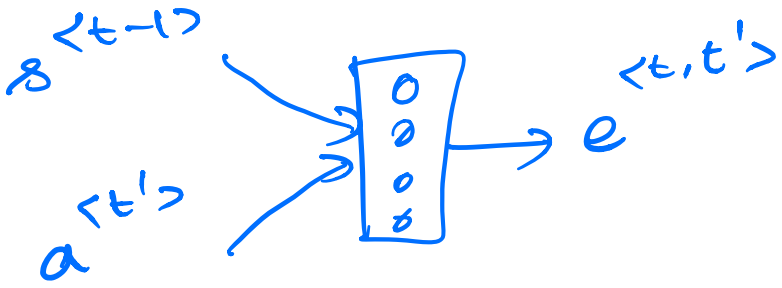
[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]

Attention model



[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]

$$\alpha^{\langle t,t' \rangle} = \frac{\exp(e^{\langle t,t' \rangle})}{\sum_{t'=1}^{T_x} \exp(e^{\langle t,t' \rangle})}$$



Time complexity : $T_x T_y$

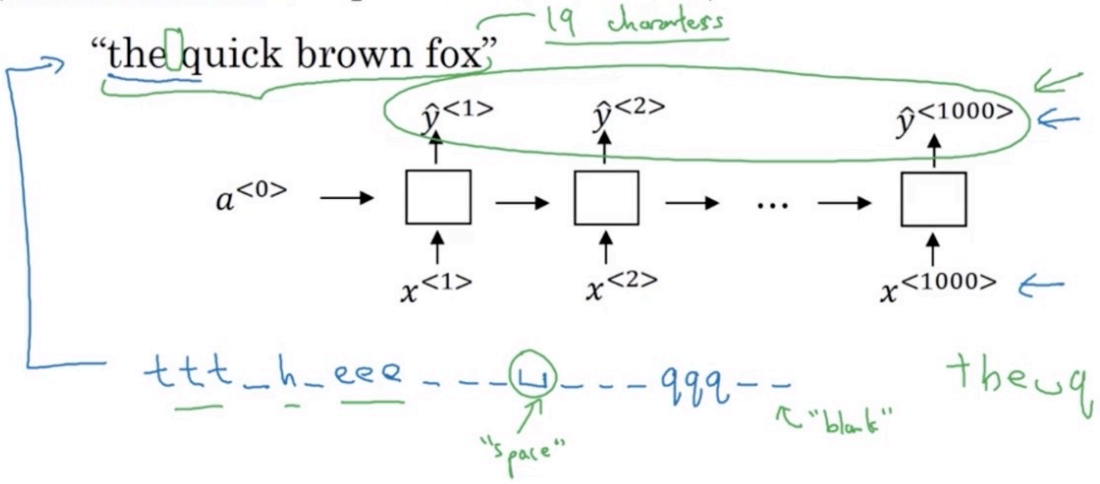
spectogram → common audio pre processing

Trigger word detection:

train RNN to output 0 when user hasn't said trigger word and output 1 when user says the trigger word.

CTC cost for speech recognition

(Connectionist temporal classification)



Basic rule: collapse repeated characters not separated by "blank"